

**Enunciado da atividade**

Continuando da **Atividade 03**, desenvolva uma aplicação em que um cliente, utilizando componentes gráficos da biblioteca **Swing**, transmita o objeto **Pessoa** (atributos: *nome* e *idade*) para o servidor, utilizando *Threads*. O servidor deve exibir em seu console de execução, os dados recebidos do cliente. O cliente deve receber um aviso do servidor, informando que os dados foram transmitidos corretamente, mostrando isso em uma **TextArea**. Você deverá desenvolver dois clientes (duas classes) para que o avaliador rode imediatamente uma classe após a outra, para fazer o uso de *Thread*.

Utilize a figura abaixo como modelo para o cliente:

**Regras para construir o programa**

A regra **número 01** desta atividade é que, na parte gráfica cliente, você deverá **centralizar a sua janela**, na tela. Haverá um desconto caso você não implemente a centralização correta da janela na tela, na ordem de **15%**. Cuidado utilizar exemplos de códigos que não centralizem a janela.

Continue usando conexão do *localhost*, ou seja, o IP **127.0.0.1**, para a conexão cliente-servidor. A porta deverá ser uma porta efêmera, **50000**. Este programa deverá ter a execução do servidor em **modo console**, e a execução do cliente em **modo gráfico**.

Não implementar nada mais do que o solicitado, tal como botões “*clique aqui para iniciar o servidor*” ou outras funcionalidades não solicitadas.

Lembre-se que o cliente não deve nunca iniciar o servidor. O servidor roda um serviço que é independente do cliente, **portanto o servidor deverá ser iniciado primeiro.**

- O seu projeto deverá ter **seu nome como autor**, em um comentário. A classe que não tiver seu nome como autor, renderá desconto global no exercício. Exemplo:

```
/*  
 *  
 * .....  
 * @author Djeizon Barros  
 *  
 */
```

- O seu projeto deverá ser entregue com a package **local.javaredes**

```
package local.javaredes;
```

O aplicativo que for entregue com *package* diferente, receberá desconto.

- Atenção, para este exercício, o servidor **NÃO DEVERÁ FECHAR A CONEXÃO**. O fechamento da conexão renderá um desconto de **15%**. Como o servidor deve receber múltiplas conexões, ele não mais deve ser encerrado automaticamente.
- Implementar ***Threads***, baseado em videoaula de apoio. **Se não houver a implementação de threads, o programa receberá a nota zero (0).**
- Fornecer **dois clientes** (duas classes, exemplo: **Cliente1.java** e **Cliente2.java**), para testes imediatos com sua aplicação.
- A *thread* deverá ser implementada **somente no servidor**, como já demonstrado na disciplina – e – **não no cliente** (embora isso também já foi demonstrado).

**Dicas importantes para concluir esta atividade**

**Dica 01:** Este exercício pode ser feito com apenas 4 arquivos. **Pessoa.java** (o objeto), **Servidor.java** (o servidor), uma classe **FormCliente1.java** e outra igual; **FormCliente2.java**. Em caso de 1 *form* para cada cliente, considere mais 2 arquivos *.form*.

**Dica 02:** Você usará *getters* e *setters* na sua classe **Pessoa.java**.

**Dica 03:** Lembre-se da criação de formulários na disciplina de **Linguagem de Programação Java I**. A IDE facilita muito a criação desses formulários, e depois, você incorpora as ações/eventos nesses campos/botões. Lembre-se dos métodos **actionPerformed** e do **AWT**.

**Regras da Entrega**

Anexe na atividade, os arquivos fonte **.java** e eventuais arquivos *.form*.

*Há alunos que conseguem desenvolver esse exercício sem a necessidade de arquivos .form, gerados pela IDE, porém se não for o seu caso, não se esqueça desses arquivos.*

- **Servidor.java** (código fonte)
- **Pessoa.java** (código fonte).
- **FormCliente1.java** (código fonte)
- **FormCliente2.java** (código fonte)
- Arquivo(s) adicional(is), se for o caso.

É permitido criar mais classes, porém, como já mencionado, **04 classes bastam** para construir esse pequeno programa.

**Não compactar nenhum arquivo.** O envio compactado, como é exigido nas outras disciplinas, **mas não nesta**, acarretará desconto.

Submeta **os arquivos** no ambiente de ensino.

**Não deixar a tarefa em Modo Rascunho.**

Clique no botão **Enviar Tarefa por Definitivo**.

**Atenção para as penalidades no exercício**

Quando o exercício recair em uma dessas situações e acumular erros, **os descontos abaixo serão aplicados**. A critério do avaliador, alguns descontos poderão ser isentados, dependendo do julgamento do prejuízo acumulado de nota.

Situação de descontos cumulativos	Desconto
Classes (ou uma delas) não possuem um comentário indicando sua autoria.	10,00
Entregou o exercício com <i>package</i> diferente.	15,00
Entregou o exercício em formato ZIP ou RAR ou outro, que é pedido nas outras disciplinas, <b>mas não nesta</b> .	15,00
Tela do cliente não centralizada; abre em um dos cantos.	15,00
Não forneceu <b>DOIS CLIENTES</b> para testes (duas classes cliente).	15,00
Foi utilizada porta baixa ou porta reservada – o programa só funciona com a troca da porta.	20,00
Foi utilizado um IP de LAN – e não um endereço do escopo 127/8 – o programa só funciona com a correção para o <i>localhost</i> .	20,00
O programa, <b>DESTA VEZ, FECHA O SERVIDOR APÓS O FINAL</b> , prejudicando o teste de <i>Threads</i> e recebimento de mais clientes.	20,00
Servidor não retorna mensagem para cliente, mas o cliente envia para o console do servidor, com sucesso.	60,00
Somente servidor funciona e entra no estado bloqueante, cliente não compila ou nem funciona.	60,00
Servidor não retorna mensagem para cliente nem retorna mensagem para o console, mas o cliente funcionou enviando o objeto.	60,00
Existe a conexão entre o servidor e o cliente, mas o objeto não é passado para o servidor, logo, o servidor não responde.	60,00
O código não compila. O código é incompreensível.	100,00
Cópia de outro aluno, não importando se tudo está correto.	100,00
O programa foi entregue, mas não há implementação de <b>Thread</b> .	100,00